

# IntelMQ hands-on workshop

TF-CSIRT/FIRST meeting Malaga 2020/1/31

Aaron Kaplan [Kaplan@cert.at](mailto:Kaplan@cert.at)



Co-financed by the Connecting Europe  
Facility of the European Union

# Overview of today

- Introduction presenters
- Check: are we ready (VMs)?
- Lesson 1: Theory section (1h)
- Hands-on Lesson 2
- Break (10:30 – 11:00)
- (Cont.) Lesson 2
- IntelMQ future
- (in parallel for the fast ones Lesson 3)

Introduction presenters

# Are we ready?

- ✓ VirtualBox + VM
- ✓ Network connectivity

Theory part

# History & background

- In the beginning, there was Abusehelper (~ 2013). Still around for some teams
  - Too complex (at that time) for us, too expensive, semi-open source, hard to get PRs upstream, etc.
  - CERT.pt (Tomas Lima) created initial IntelMQ version in ~ 2014
  - “works”
  - Aaron Kaplan brings in IntelMQ to CERT.at, CERT.at takes over maintainership
  - Many steps to change the PoC to production ready code
  - **Emphasis on open source (via github.com)**
  - **Emphasis on KISS**

# The IntelMQ promise

IntelMQ follows the following basic meta-guidelines:

- Don't break simplicity – **KISS (think Lego blocks for IT security automation)**
- Keep it open source - forever
- Strive for perfection while keeping a deadline
- Reduce complexity/avoid feature bloat
- Embrace unit testing
- Code readability: test with unexperienced programmers
- Communicate clearly

# The community project

- At the heart of all of this is: **IntelMQ is a community project**
- Open source for ever (AGPL v3)
- Many contributions (thanks to CSIRT.cz, .SK, .PT, BSI/CERT-Bund, etc. etc.)
- Maybe one day, you will join us?
- It's easy if you try



# The extended community project - IHAP

- Incident **H**andling **A**utomation **P**rojects
- Regular meetings of tool developers for incident handling automation
- Next meeting in Vienna ~ April 2020 (TBA)
- <http://www.enisa.europa.eu/activities/cert/support/incident-handling-automation>
- **Mailing-list:** [ihap@lists.trusted-introducer.org](mailto:ihap@lists.trusted-introducer.org)
- Subscription? → ping [Kaplan@cert.at](mailto:Kaplan@cert.at)

# CERT.at's role in IntelMQ

- Maintainer / steward of the project
- Contribute code
- Code submission QA review
- Release mgmt
- Coordination
- Architecture design of future versions (with the community)

Okay, so what is IntelMQ?

# TL;DR Version (1)

- A framework (python) of “**lego blocks**” of IT security CERT automation.
- Lego blocks are **simple (KISS principle)**.
- Lego blocks can be re-combined as you need.
- You are missing a block? -> Look, if someone already wrote it. If not, write it and share with the community → synergy effects
- Every CERT has its own workflow.
- Blocks can be connected with each other to create “flow” via a MQ
- Data-flow oriented architecture
- The stuff that “flows” is log lines (“events”)
- Similar to unix pipelines

Collector

Parser

Expert

Output

Edit Defaults

Add Bot

Add Queue

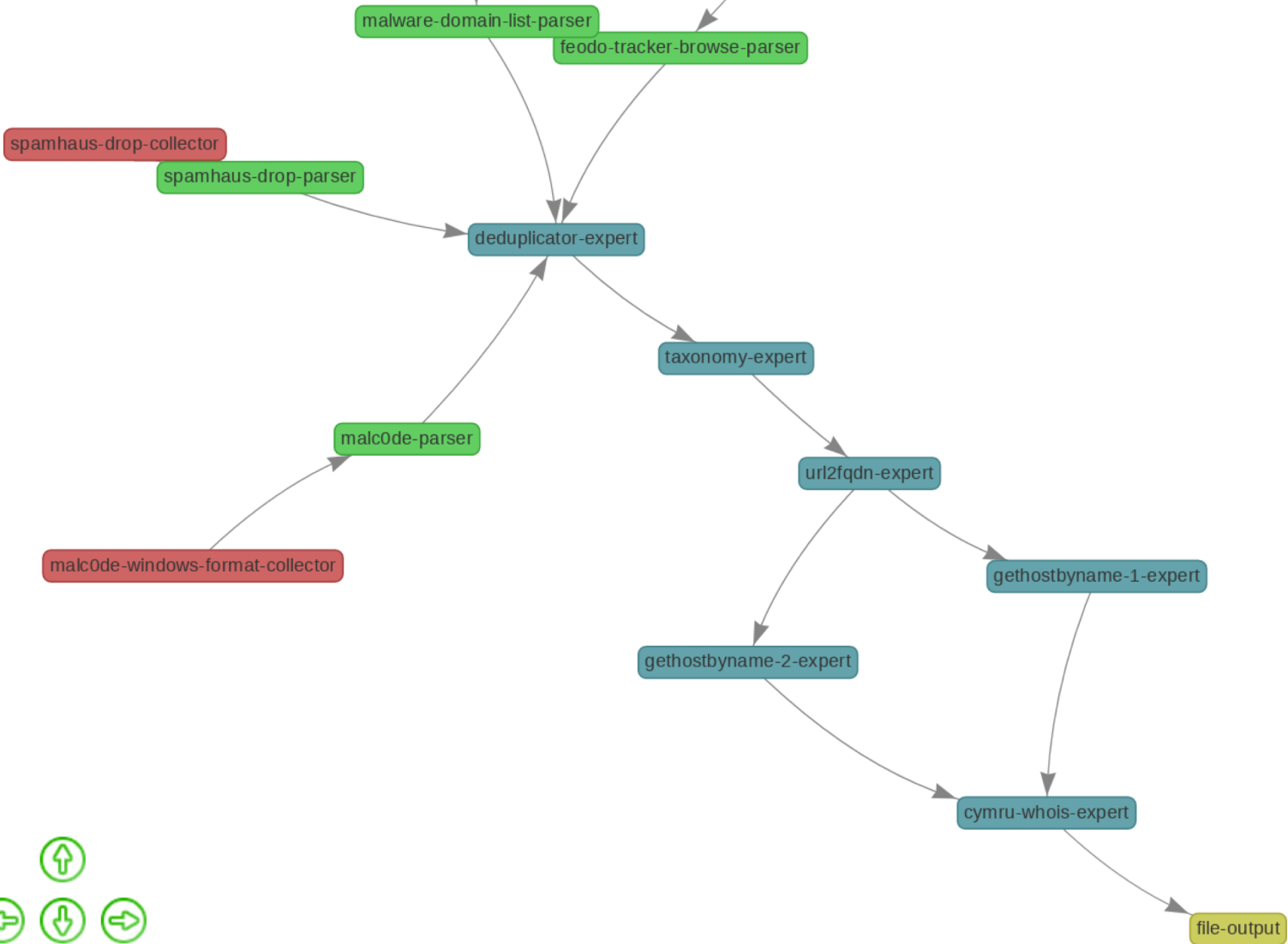
Live

Physics

Redraw Botnet

Clear Configuration

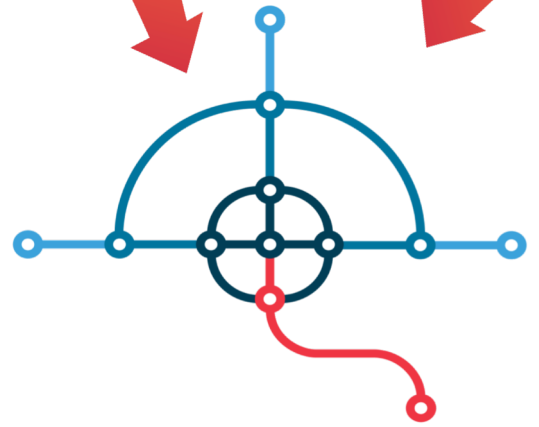
Save Configuration



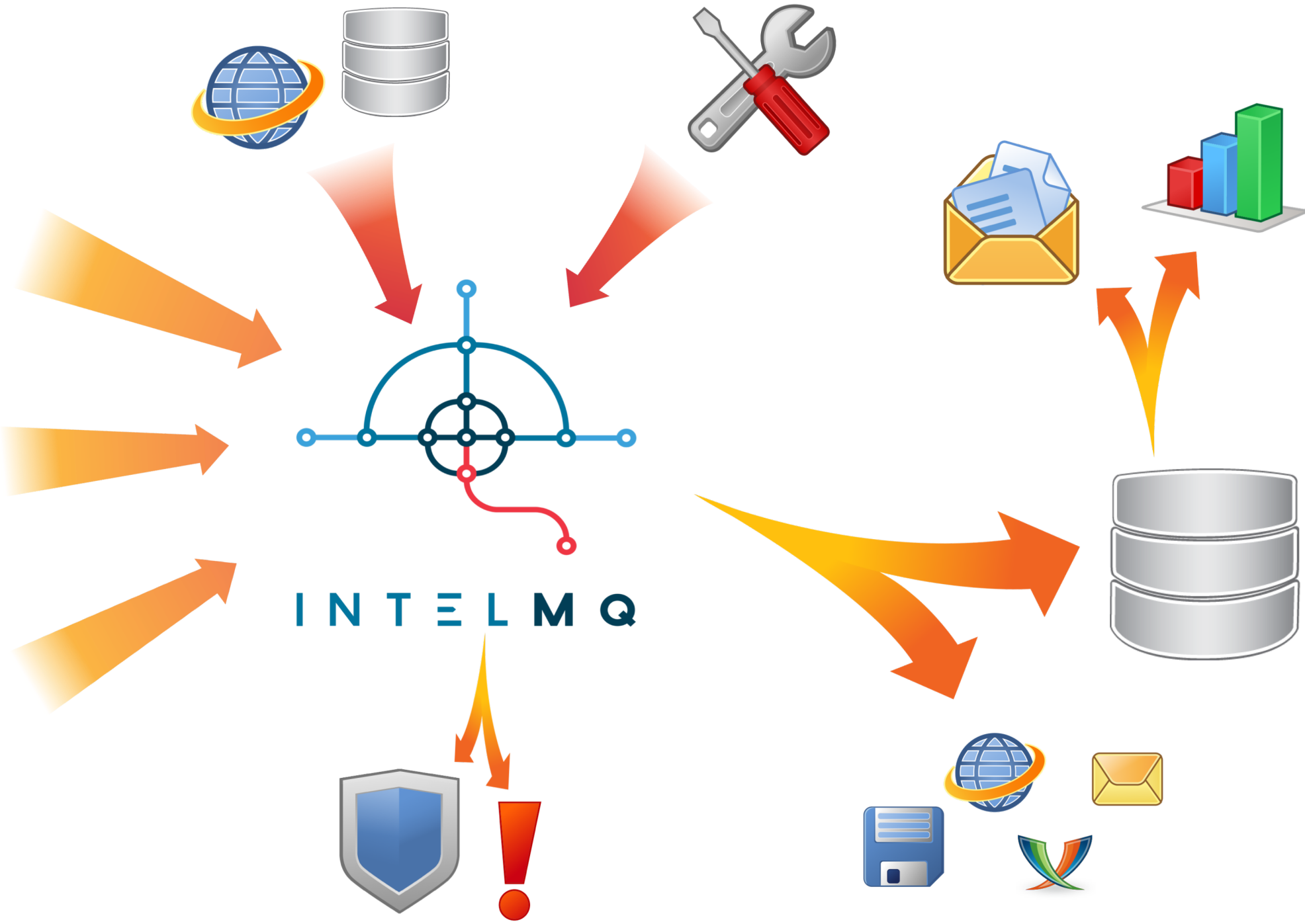
normal

# Use-cases: what do we need to solve?

- A CERT receives tons of data feeds (shadowserver, etc.)
- There is no single format for all of that → write parsers for everything(?)
- Need to process, filter, verify, ...
- send out to constituency and/or
- React on the data feed (firewall blocks, etc.)
  
- → IntelMQ is the **glue** for this streaming data. It connects.



INTELMQ



# Differentiation from MISP

- MISP was meant to
  - share IoCs amongst analysts (especially APTs etc)
  - **Correlation** is a key feature
  - Lower volume
- IntelMQ was made to as an ETL (Extract Transform Load) framework for
  - High throughput / high volume
  - No Correlation
  - For parsing all the feeds, a CERT might receive , process it a bit and send it on.



# Terminology

- “feed”
  - Streaming or
  - Download
- “Bot”
- “Botnet”
- “Parser”
- “Collector”
- “Expert”
- “Output”
- “Report”
- “Event”
- “DHO” – Data Harmonization Ontology (== internal format for events)

# Terminology in Detail

- Bot = Small python script which
  - Inherits from the Bot class
  - Implements an init() method
  - Implements a process() method
- Botnet = a collection of bots. A set of DAGs of bots.
- Pipeline = the structure of the MQs which connect the bots

# Example bot

```
class MyBot(Bot):  
    def init(self):  
        # optional initialization  
    def process(self):  
        event = self.receive_message() # dict  
        # process event  
        self.send_message(event)
```

Collector <

Parser <

Expert <

Output <

Edit Defaults

+ Add Bot

+ Add Queue

Live

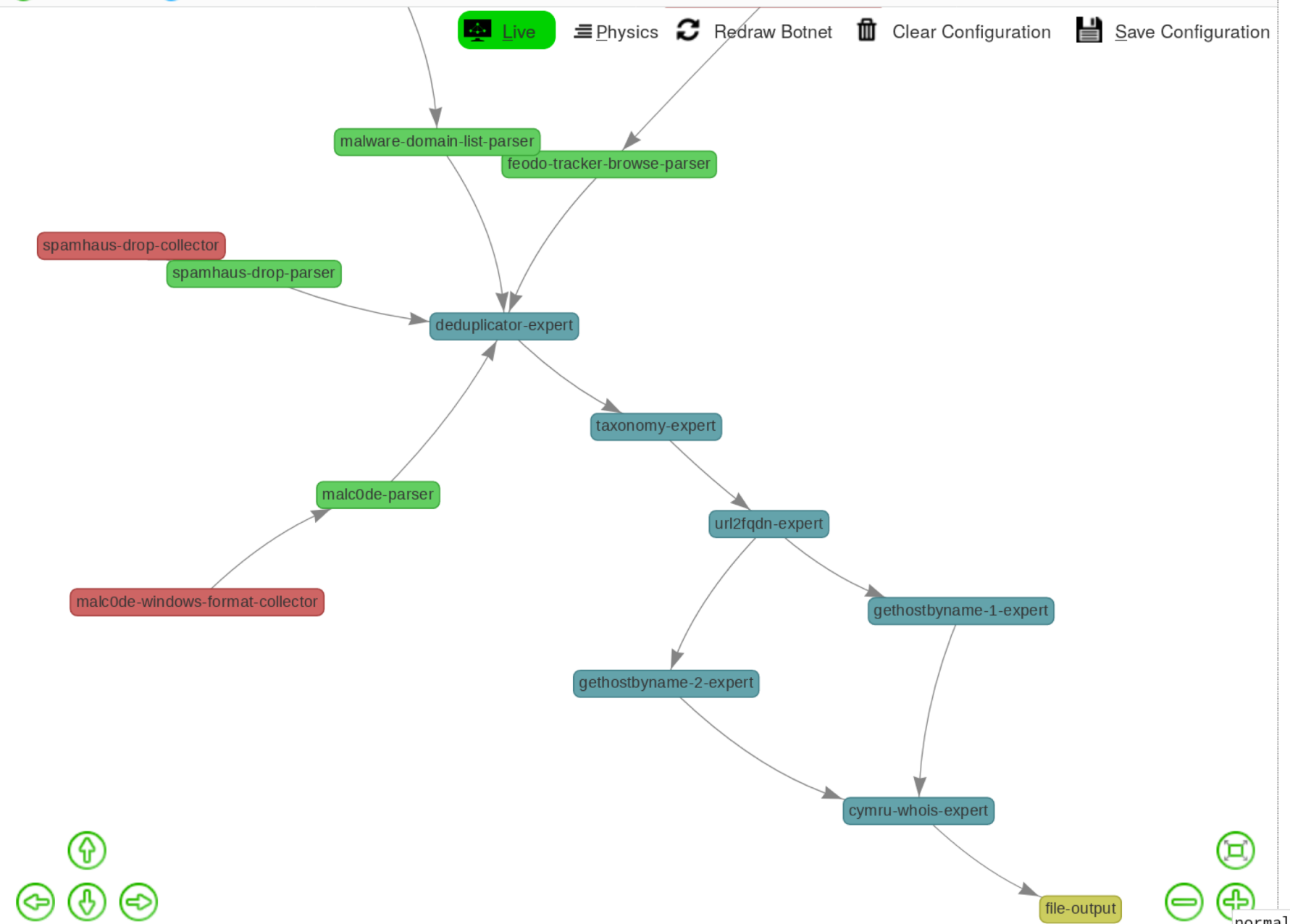
Physics

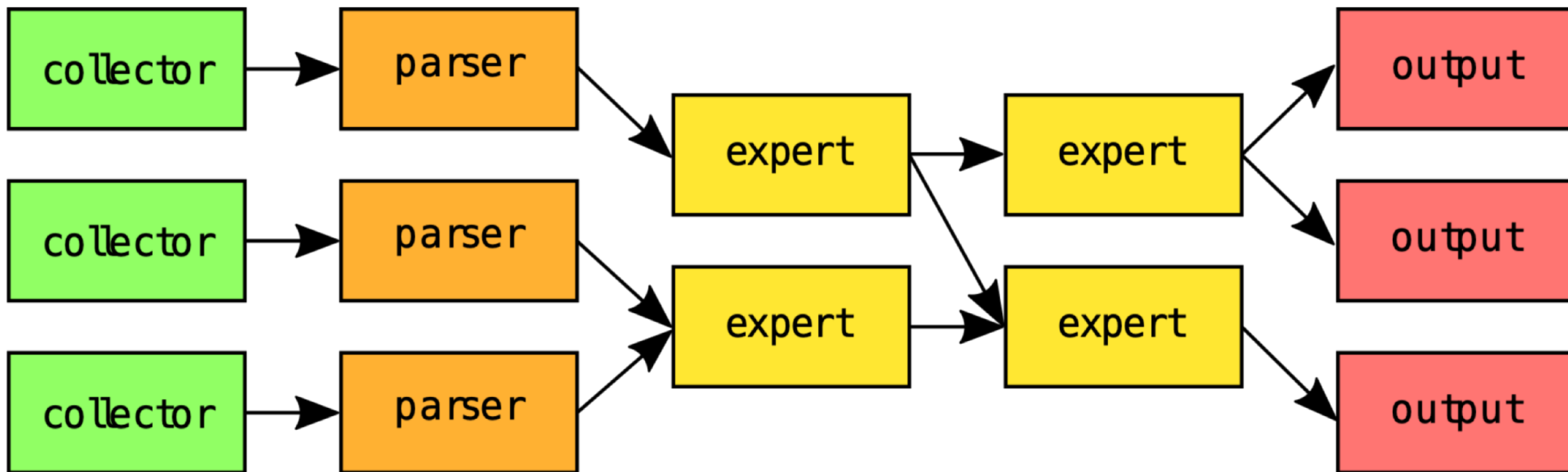
Redraw Botnet

Clear Configuration

Save Configuration

# Pipelines





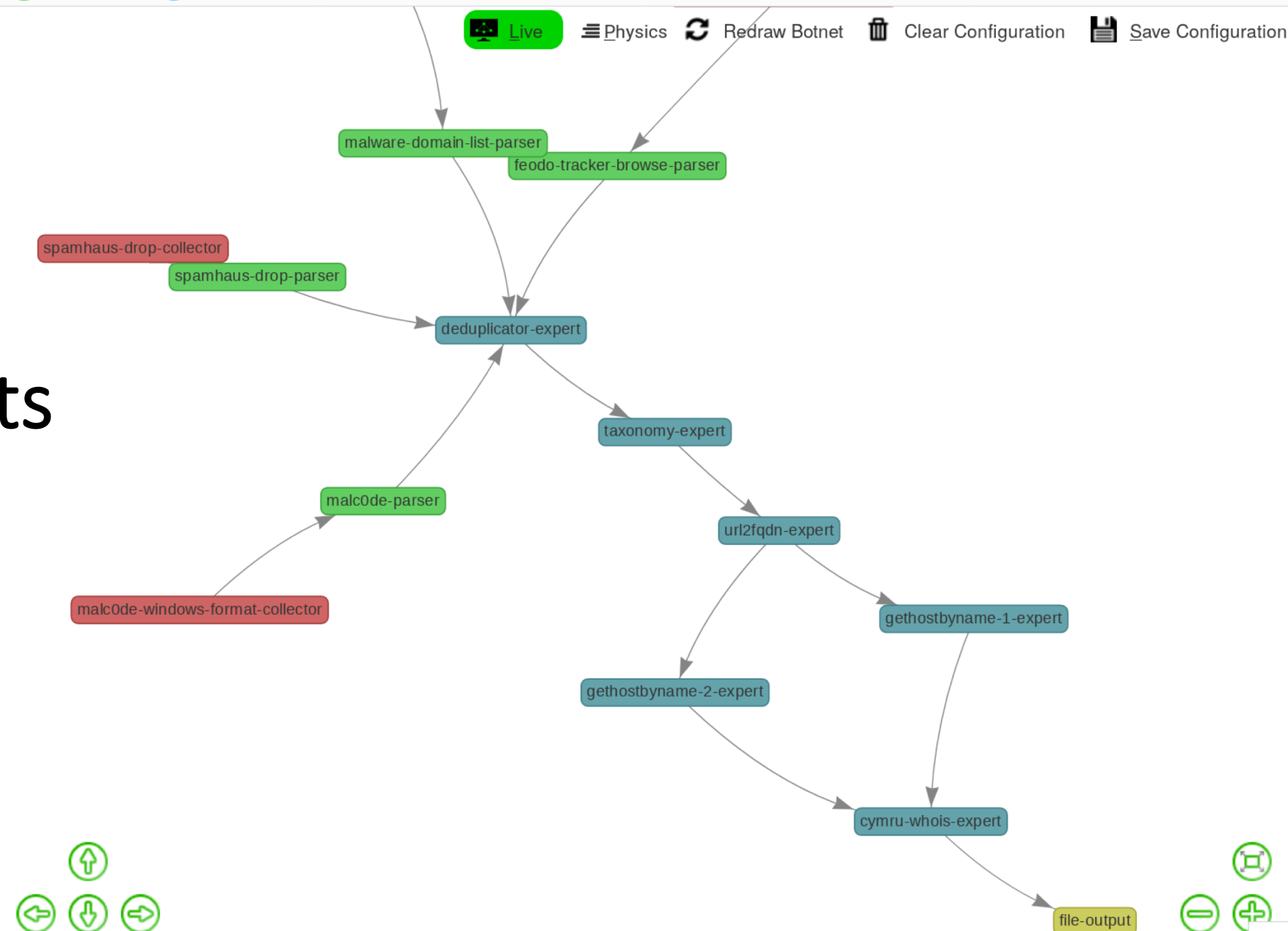
# Terminology: types of Bots

- Collector → emits Report
  - Parser → emits Events
  - Expert → enriches / filters
  - Output → sends it somewhere
- 
- Examples for Experts:
    - filter by country code (expert)
    - Add ASN by IP address
  - Examples for Output:
    - Write to MongoDB, PostgreSQL DB
    - Write to Elastic Search
    - Send to Ticket System
    - Send via email

- Collector
- Parser
- Expert
- Output

Edit Defaults

Buttons: Add Bot, Add Queue, Live, Physics, Redraw Botnet, Clear Configuration, Save Configuration



# Types of bots

Navigation icons: up, down, left, right arrows

Navigation icons: zoom in, zoom out, normal

# Types of Messages

- Report:
  - Data (base64 combined) + metadata
  - Example: the whole blacklist from spamhaus
- Event:
  - A report gets split up into individual log lines (== “Events”)
  - An event is **in the DHO format**
  - **Following bots in the pipeline can rely on the format**



# DHO

			retrieval/generation.
Source	source.abuse_contact	LowercaseString	Abuse contact for source address. A comma separated list.
Source	source.account	String	An account name or email address, which has been identified to relate to the source of an abuse event.
Source	source.allocated	DateTime	Allocation date corresponding to BGP prefix.
Source	source.as_name	String	The autonomous system name from which the connection originated.
Source	source.asn	ASN	The autonomous system number from which originated the connection.
Source	source.domain_suffix	FQDN	The suffix of the domain from the public suffix list.
Source	source.fqdn	FQDN	A DNS name related to the host from which the connection originated. DNS allows even binary data in DNS, so we have to allow everything. A final point is stripped, string is converted to lower case characters.

- <https://github.com/certtools/intelmq/blob/develop/docs/Harmonization-fields.md>

# Configuration parameters

```
© Perform Action... >_ Command 62% 12 GB 0.0 k
user@malaga:~$
user@malaga:~$
user@malaga:~$ cd /opt/intelmq/etc/
user@malaga:/opt/intelmq/etc$ ls -al
total 180
drwxrwxr-x 3 intelmq www-data 4096 Jan 30 07:38 .
drwxr-xr-x 4 intelmq intelmq 4096 Jan 30 07:38 ..
-rw-rw-r-- 1 intelmq www-data 51899 Jan 31 08:59 BOTS
-rw-rw-r-- 1 intelmq www-data 1328 Jan 31 08:59 defaults.conf
-rw-rw-r-- 1 intelmq www-data 68603 Jan 31 08:59 feeds.yaml
-rw-rw-r-- 1 intelmq www-data 21085 Jan 31 08:59 harmonization.conf
drwxrwxr-x 2 intelmq www-data 4096 Jan 23 17:11 manager
-rw-rw-r-- 1 intelmq www-data 2351 Jan 31 08:59 pipeline.conf
-rw-rw-r-- 1 intelmq www-data 8649 Jan 31 08:59 runtime.conf
-rw-r--r-- 1 root root 466 Jan 23 17:11 webinput_csv.conf
```

# Configuration parameters

- JSON files
  - Runtime.conf – bots parameters
  - Pipeline.conf – the pipeline setup

Questions on the theory?